Deploying Wireless Sensor Networks with Fault-Tolerance for Structural Health Monitoring

Md Zakirul Alam Bhuiyan, *Member*, *IEEE*, Guojun Wang, *Member*, *IEEE*, Jiannong Cao, *Senior Member*, *IEEE*, and Jie Wu, *Fellow*, *IEEE*

Abstract—Structural health monitoring (SHM) systems are implemented for structures (e.g., bridges, buildings) to monitor their operations and health status. Wireless sensor networks (WSNs) are becoming an enabling technology for SHM applications that are more prevalent and more easily deployable than traditional wired networks. However, SHM brings new challenges to WSNs: engineering-driven optimal deployment, a large volume of data, sophisticated computing, and so forth. In this paper, we address two important challenges: sensor deployment and decentralized computing. We propose a solution, to deploy wireless sensors at strategic locations to achieve the best estimates of structural health (e.g., damage) by following the widely used wired sensor system deployment approach from civil/structural engineering. We found that faults (caused by communication errors, unstable connectivity, sensor faults, etc.) in such a deployed WSN greatly affect the performance of SHM. To make the WSN resilient to the faults, we present an approach, called FTSHM (fault-tolerance in SHM), to repair the WSN and guarantee a specified degree of fault tolerance. FTSHM searches the repairing points in clusters in a distributed manner, and places a set of backup sensors at those points in such a way that still satisfies the engineering requirements. FTSHM also includes an SHM algorithm suitable for decentralized computing in the energy-constrained WSN, with the objective of guaranteeing that the WSN for SHM remains connected in the event of a fault, thus prolonging the WSN lifetime under connectivity and data delivery constraints. We demonstrate the advantages of FTSHM through extensive simulations and real experimental settings on a physical structure.

Index Terms—Wireless sensor networks, deployment, fault-tolerance, energy-efficiency, structural health monitoring

1 INTRODUCTION

The new advances in sensor device technologies make wireless sensor networks (WSNs) effective and economically-viable solutions for a wide variety of applications, such as environmental monitoring, scientific exploration, and target tracking [1]–[4]. Civil structures, including bridges, buildings, tunnels, aircrafts, nuclear plants, among others, are complex engineering systems that ensure society's economic and industrial prosperity [5]–[9]. Structural health monitoring (SHM) systems are implemented for these structures to monitor their operations and health status. WSNs are becoming an enabling technology for SHM that are more prevalent and more easily deployable than current wired systems. Examples include the Golden gate bridge in the US [5], bridge monitoring in India [7], and Guangzhou new TV tower (GNTVT) in China [10].

- *M.Z.A. Bhuiyan is with the School of Information Science and Engineering, Central South University, Changsha 410083, China, and Department of Computing, the Hong Kong Polytechnic University, Kowloon, Hong Kong. E-mail: zakirulalam@gmail.com.*
- *G. Wang is with the School of Information Science and Engineering, Central South University, Changsha 410083, China. E-mail: csgjwang@mail.csu.edu.cn.*
- *J. Cao is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: csjcao@comp.polyu.edu.hk.*
- *J. Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122. E-mail: jiewu@temple.edu.*

Manuscript received 17 Mar. 2013; revised 10 Sep. 2013; accepted 17 Sep. 2013. Date of publication 30 Sep. 2013; date of current version 16 Jan. 2015. Recommended for acceptance by H. M. Ammari. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TC.2013.195

The objectives of SHM are to determine health status (i.e., damage, which is a remarkable change around a sensor location) of a structure, and provides both long-term monitoring and rapid analysis in response to unusual incidents, e.g., earthquakes, load, etc. In practice, it is often difficult to achieve these objectives in WSN-based SHM, due to requirements of SHM and severe limitations of WSNs [11], [12].

One of the fundamental requirements of SHM is sensor location optimization. According to deployment methods from civil/structural/mechanical engineering, wired sensors are usually deployed at strategic locations to achieve the best estimates of structural health status [11], [13]–[15]. These methods do not support sensor deployment anytime or anywhere in the structure. They also require significant domain knowledge along with SHM complexity. Generic random, uniform, or grids-based WSN deployments may not be suitable for SHM. Bearing these in mind, we first focus on *deploying a set of N wireless sensors (called primary sensors) in a set of locations of a structure by using the most widely accepted sensor location optimization method* from the engineering domains, EFI (EFfective Independence) [13], [16], [11].

Since generic WSN deployments are not bounded by the engineering-like requirements, locations of sensor or relay nodes can be planned to prolong network lifetime by ensuring connectivity and reliable data delivery. In contrast, a notable fact in SHM is that once a set of N sensors are deployed (no matter if N is a few or many) and analysis of structural physical properties [11] is carried out by a base station (BS), data from each sensor location must be collected for SHM. Thus, at first sight, wired networks seem a more reliable and stable choice than WSNs. There are severe constraints in

WSNs, such as error-less communication, fault tolerance, energy, bandwidth, etc. There are more chances that the deployed WSN for SHM is prone to faults (e.g., getting separated into multiple components) for various reasons: (1) physical structural modeling constraint; (2) irregular communication or unstable connectivity; (3) sensors' debonding faults; (4) quick energy depletion of some sensors (they may only be the points by which other sensors transmit data); (5) there is irregular communication distance—transmitting data from a sensor to another sensor, or the BS over large structures is not reliable.

If any of the fault types occur in WSNs, two problems arise: *how to continue obtaining monitoring information* and *how to guarantee sensor fault tolerance in SHM*. Without these answers, we are unable to know at some moment: is a structure going to crash? The fault tolerance problem has been studied extensively in diverse applications of WSNs by researchers in the computer science (CS) community. This is ignored in the SHM applications.

We consider the problem of detecting possible repairing points (RPs) in the WSN. We present an approach, called `FTSHM` (fault-tolerance in SHM), to repair the network before it starts operations, so as to guarantee a specified degree of fault-tolerance. `FTSHM` searches the repairing points or locations in clusters, and places a set of *backup sensors* at those points by satisfying engineering requirements. In fact, searching the RPs is a prediction of future network failure points (e.g., separable points, isolated points, and critical middle points), which is a promising idea (to search such points and tackle them in advance). To search highly possible RPs, we think of this searching in a distributed manner: it involves only local communication between neighbors in a cluster, and limits searching to clusters (i.e., cluster by cluster).

Another fundamental requirement of SHM is structural damage identification. Existing SHM algorithms work on the raw data of multiple sensors at a high frequency (X00 times per second, $X = 1, 2, \ldots$). Each sensor works actively for a long period of time, say, from 10 minutes to hours, subjected to severe constraints on radio bandwidth and energy usage. Given these constraints, it is typically not possible to acquire data continuously from all nodes in a global BS [5], [17], [18]. As a result, SHM applications strive to acquire the most "interesting" data (e.g., when there is an event of an earthquake or of damage in a structure) while wasting resources on "uninteresting" data [19]. To prolong the WSN lifetime, the energy cost of each sensor for monitoring must be carefully considered. We present an energy-efficient SHM algorithm, called *Damage-Indicator*. This runs on each sensor and then provides a light-weighted indication of damage in a cluster in a decentralized manner. If there is no indication found in the cluster, the "uninteresting" data transmission toward the BS can be reduced.

The major contribution of this paper is four-fold:

- We formulate the problem of placing a small set of backup sensors into a deployed WSN with primary sensors, and design the `FTSHM` to address the problem, which is no easy task, as it incorporates multi-domain knowledge.
- To make the WSN resilient to the faults, we propose a backup sensor placement (BSP) algorithm that includes several sub-algorithms.
- To make the resource-constrained WSN easier to use for SHM, we propose an SHM algorithm, Damage-Indicator, showing how a traditional centralized SHM framework can be transformed into a decentralized one.
- We conduct a comprehensive evaluation of `FTSHM`. In simulation studies, we use data sets collected from the GNTVT system (a SHM project of Hong Kong PolyU) [10]. In a real-world deployment, we utilize integrated Imote2 sensors that run on TinyOS. The effectiveness of `FTSHM` is compared with that of existing approaches.

This paper is organized as follows. In Section 2, we discuss related work. Section 3 describes engineering-driven deployment methods and their limitations. We provide system models and formulate the problem in Section 4. The proposed BSP algorithm is given in Section 5. Section 6 offers the proposed Damage-Indicator algorithm. Evaluation via simulations and real deployments are described in Sections 7 and 8, respectively. Finally, Section 9 concludes this paper.

2 Related Work

Indeed, WSNs are gradually becoming prevalent for SHM applications, focusing on reliable collections of raw signals at relatively high sampling rates, synchronization, and so on [3], [6], [8], [12], [15], [18], [20], [21]. Interestingly, the problem of sensor deployment for SHM is seldom considered, although it is one of the fundamental requirements of SHM [11], [14]. Here, we only discuss the sensor deployment perspective. We discuss the WSN-based SHM systems' perspectives in Appendix A of the supplemental material, which can be found in the Computer Society Digital Library at https://doi.ieeecomputersociety.org/10.1109/TC.2014.195.

Various optimal sensor deployment methods from engineering domains have been advanced for wired network systems [11], [13], [14], [16], [20], [22]. Some WSN systems are deployed and verified with an interest in knowing whether a WSN for SHM system can replicate the data collection functionality of the original wired-based counterpart. Those systems sometimes choose powerful sensors to accomplish tasks that could have been achieved by more cost-effective counterparts, through system optimization. They still have difficulties in handling the WSN constraints. They also are not very concerned with the effects of sensor faults, transmission faults, WSN separation, etc., on a deployed WSN on a structure.

Based on the wired sensor deployment feasibility, a noteworthy work on WSN deployment, SPEM, is suggested and is also verified on the GNTVT [11]. The main idea of SPEM is to adjust the sensor locations of the structure, using an engineering method to better fit both engineering and CS requirements. However, we found that such adjustments (with a reduction of location quality for sensor placement) may lead to the loss of some optimal locations. This may be inappropriate for SHM. In addition, it is fully centralized and may not assure the quality of SHM, if a fault on a sensor (which is placed at an optimal location) or on a packet transmission occurs. Moreover, some sensors work as relays for many other sensors to support data collection at the BS. They may fail during operation.

It can be seen that WSN deployment for monitoring a structural event (e.g., damage, crack, etc.) is not as straightforward as in other applications. With the generic WSN deployment methods, effective SHM may not be possible. This is because the spatial information to describe the sensitivity of an event is not sufficient at many locations, where monitoring damage is due largely to structural location sensitivity. For example, existing sensor deployment (for habitat monitoring, target tracking, and so on) with grids, or at intersection points [23]–[25], may not be meaningful for SHM. Optimal deployment for k-coverage [26] and additional/relay node deployment can be seen in [27]–[29]. The relay deployment ('RELAY' for short) [29] considers both uniform and random distribution for data collection. The idea is to deploy a set of sensors, which collect sensing data and another set of sensors, to relay the data from the sensors in the first set to the BS. They may not satisfy SHM requirements.

The proposed FTSHM attempts to overcome the limitations above by combining engineering requirements, e.g., location quality, mode shape analysis (see Definition 1), with CS requirements, e.g., strong connectivity, low communication cost, and fault tolerance. FTSHM justifies the impact of WSN faults on SHM. Also, the algorithm in FTSHM, Damage-Indicator, reduces the communication burden on the resource-constrained WSN, resulting in a prolonged lifetime.

3 Engineering-Driven Deployment Methods and Their Limitations

In this section, we begin with one of the important structural properties needed for SHM: mode shape. Then, we describe the engineering-driven methods and their limitations. Finally, we highlight how many backup sensors can be deployed to improve them.

3.1 Structural Mode Shapes

Definition 1 [Φ: Mode shape]. *Each mechanical structure has a number of specific vibration patterns at specific frequencies. These vibration patterns are called mode shapes.*

Definition 2 [Mode]. *A pattern of vibration under ambient or forced excitations, which is numbered according to the number of half waves in the vibration.*

Definition 3 [Reference Mode Shape]. *A reference mode shape is a mode shape estimated by using a set of measured vibrations when the structural health condition is normal or undamaged. Reference global mode shapes are calculated by the BS, and correspond to the centralized data acquisition and processing, commonly used in the wired systems.*

An essential problem in SHM is the sensor location optimization. This problem consists of an arrangement of a limited number of sensors over the structure that guarantees the best estimates of the structural properties, such as mode shapes. Mode shapes and their derivatives have been proven to be very sensitive in capturing structural dynamic changes. A significant change in the mode shape implies possible damage. By comparing changes, and by identifying the sensor location where the maximum changes occur, we can obtain possible damage-sensitive locations

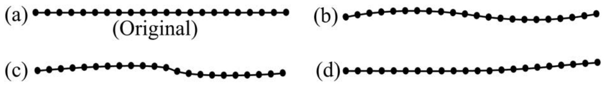

Fig. 1. Mode shapes of the LSK building structure: (a) original mode; (b) mode 1; (c) mode 2; (d) mode 3.

$$[\Phi_1 \quad \Phi_2 \dots \Phi_p] = \begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1p} \\ \phi_{21} & \phi_{22} & \dots & \phi_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{m1} & \phi_{m2} & \dots & \phi_{mp} \end{bmatrix}. \tag{1}$$

Using (1), we estimate the kth mode shape, denoted by Φ_k of a structure, captured by ambient vibration frequencies [12]. A sensor can capture the frequencies. For example, we deploy a total of m sensors on a structure and extract a total of p mode shapes, where $\Phi_k = [\phi_{1k}\ \phi_{2k}\ \dots\ \phi_{mk}]'$ is the kth vibration pattern of the structure. $\Phi_{ik}(i = 1, 2, \dots, m)$ is the kth mode shape value, defined at the ith sensor. For example, Fig. 1 exhibits the first three mode shapes of the structure, extracted from the measurements of 22 deployed sensors. Φ_k has an element corresponding to each sensor.

3.2 Sensor Deployment Procedure

There are several optimal sensor deployment methods available in engineering domains, e.g., effective independence (EFI), EFI driving-point residue (EFI-DPR), kinetic energy method (KEM), and so on [13], [14]. We use EFI for our sensor placement, which is a widely accepted method from the domains. EFI uses mode shape and noise measurement, and then provides the Fisher information matrix (FIM) determinant, which helps calculate EFI values. The value is referred to as the placement quality indicator. The larger the value, the better the placement. An iterative algorithm is used [14] to evaluate the candidate location of each ith sensor based on EFI values, where each value corresponds to a candidate location. Suppose that there are M candidate locations in a structure. M is arbitrarily large. All M locations are sorted by respective EFI values. A sensor location with the smallest value is eliminated. The iteration continues until the deployment of given sensors is done.

3.3 Limitations with These Deployment Methods

The methods, EFI or others, are mainly used for wired network systems. Such systems are not usually prone to faults, even though in many cases, there is no need to tackle faults. However, there are more chances for a WSN deployed through such a method to be prone to faults, due to various reasons:

- Structural modeling constraint: Wireless sensor deployment at some strategic locations provide the best monitoring results, but the results may not reach the BS efficiently. This is due to the physical structural modeling [20]. In a real-world SHM, such locations may be inaccessible in some cases: (i) due to the proximity to the bearing load zone (e.g., in a bridge structure) or environmental effect (e.g., heavy wind); (ii) irregular sensing field, e.g., not a square or circle-shape on structures, like a bridge, building, etc.

- Some sensors often have irregular communication or unstable connectivity with others, due to reasons such as the location not being suitable for wireless sensor node communication, the environmental interference being very high, etc.
- Some sensors may show debonding faults, mechanical degradation, etc.
- Sensor failure is expected to be quite common, due to the typically-limited energy budget. There are some sensors that deplete their energy more quickly than others, because of huge transmission. Also, they may be the points by which many other sensors transmit data. Failure of such a sensor reduces the number of multi-hop paths in the network. Such failures can cause a subset of nodes (those that have not failed) to become disconnected from the rest of the network.
- Irregular communication distance: Imagine that we are given a large civil structure for monitoring, e.g., the GNTVT, or Ting Kao bridge [9]. Having large communication distances between two sensors, and between sensors and the BS, is a key difficulty for data transmission.

3.4 The Number of Sensors to Be Deployed

A question may arise: *how many backup sensors do we need to place* after deployment of N (primary) sensors? Suppose that we are given a limited number of sensors (m) for deployment, and are required to fulfill two conditions. First, sensors should be deployed at optimal locations, since SHM applications do not support deploying sensors anywhere. Second, we should guarantee a specified degree of fault tolerance before the WSN formally starts operations.

Therefore, we plan to place a small number of backup sensors so that we can repair possible failure points in a WSN in advance. Let N and R be the number of primary and backup sensors, respectively. Both N and R are due to the SHM user's plan. We then consider that our FTSHM approach could be one of two versions: 1) $m \geq N + R, N > R$, i.e., we can separate a number of sensors as backup sensors from the given sensors; 2) $N(= m) + R(< N)$, i.e., we are given two separate sets of sensors. In this case, a large number of sensors may bring extra burden on the WSN, besides degrading the quality of monitoring. This situation will be more serious if it is a centralized/global SHM system. In FTSHM, we address the first version, i.e., after deployment of N sensors. R is placed in a subset of near-optimal remaining/unused locations around the repairing points.

4 Problem Formulation and Models

Consider that a set P of N primary sensors, and a set B of R backup or redundant sensors, are to be deployed for SHM. A structure F has a set of M feasible locations for sensor placement. Consider that *N primary sensors are deployed on F by finding candidate locations out of M, using EFI values* [11]. After deployment, N sensors can be connected through clusters in a single to multi-hop WSN. The location of the BS is predetermined, and suitably located, but may be far away from the sensing structure. For simplicity, we assume that the deployed WSN with N primary sensors, $P = \{l_i = (x_i, y_i) \in F, 1 \leq i \leq N\}$, is prone to faults and is weakly connected (i.e., k-connected, $(k \geq 1)$), where sensor i is placed at l_i. It is highly possible that the network connectivity is unstable in the structural environment. It is crucial in SHM, once a set of sensors are deployed, that data must be collected from each optimal location for SHM. Thus, data delivery must be fault-tolerant in the WSN. Therefore, our original problem is to place a set B of R backup sensors to achieve a fault-tolerant WSN.

4.1 Communication Model

To compute the energy consumption denoted by E, we use a well-accepted transmission model [30]. This assumes that the power per bit for transmission over a wireless link is a function of the distance (d_{uv}) between a transmitter (u) and a receiver (v). A similar power model has been widely adopted to study various theoretical aspects of WSNs [31], [32]. If the distance is d_{uv}, the required power is given by

$$E \propto d_{uv}^{\kappa}, \tag{2}$$

where κ is the path-loss exponent (usually, $2 \leq \kappa \leq 4$). Also, we consider each sensor as having a maximum $(E_{\max})$ and a minimum power level ($E_{\min}$), at which it can transmit. We put a limit on R_{lr} (the maximum or long transmission radius), $d_{uv}^{\kappa} \leq E_{\max}$. Thus

$$d_{uv} \leq R_{lr}, \tag{3}$$

where $R_{lr} = (E_{\max})^{\frac{1}{\kappa}}$. Basically, R_{lr} is used by a cluster head (CH) (will be described later) in FTSHM; otherwise, a sensor uses R_{sr} (the minimum or short transmission radius). We adopt these two types, because frequent longer radius or direct communication in the WSN unnecessarily drains the batteries of sensors, eventually resulting in a fault in the packet transmission, or a sensor failure. We have to ensure that sensor deployment for SHM is reliable in data transmission, and the amount of unnecessary retransmission is minimum. Similarly, communication cost depends on both the sensor deployment (which determines locations of sensors) and the transmission strategy (which determines how the sensors are connected and transmit data).

Besides the power required for communication between adjacent sensors, a communication metric of importance is the quality of a wireless link. We estimate an acceptable link quality, denoted by $c_T(\{u, v\})$, using an existing link model (see Appendix B of the supplemental material for more details). $c_T(\{u, v\})$ confirms having the sensor placement at communication-efficient locations.

4.2 Data Delivery

We use a simple, but energy-related cost function for data delivery to study the interplay between sensor deployment for SHM and data transmission. The total cost of data delivery (sensors are rooted at CHs, all CHs are rooted at the BS) denoted by D_d, is given as

$$D_d = \sum_{i}^{m} D_i(x) \times c_i \quad i = 1, 2, \ldots, m, \tag{4}$$

where $D_i(x)$ is the amount (x[bit]) of data transmitted by the ith sensor, and c_i is the per-bit transmission cost from any sensor i to its CH or sink. D_d is due to the amount of x[bit] for d_{uv}.

4.3 Network Lifetime (T)

Consider that the energy consumption is E_m for taking measurements, and E_c for computation of the final mode shape or damage indication on each ith sensor. Then, the total energy consumption of each sensor i for a round of monitoring is given by

$$E_i = E + E_m + E_c. \tag{5}$$

Let $E_i^t = \max_{i=1,2,\ldots,m} E_i$, which is the maximum total energy consumption on the ith sensor. Let E_r be the residual energy on the ith sensor. We define the system lifetime T to be the total rounds of monitoring before any battery runs out of energy [31]

$$T = E_r / E_i^t. \tag{6}$$

4.4 Problem Definition

The problem is to place a set B of R backup sensors into a WSN with N primary sensors by optimally finding locations out of $(M - N)$ remaining locations such that: (i) the WSN is guaranteed to be fault-tolerant to the presence of up to $k-1$ sensor faults and data packet-losses, and (ii) T is prolonged under constraints (3), (4), $c_T(\{u,v\})$, and k-connectivity.

5 BSP: Backup Sensor Placement

In this section, we first briefly describe an SHM application-specific clustering. Then, we propose our BSP algorithm, including several sub-algorithms.

5.1 Clustering

In FTSHM, we detect possible repairing (failure) points in the WSN, and repair them by placing backup sensors through clusters. We improve an existing clustering approach suggested in [12] for SHM, called C-SHM, which is specifically designed for SHM application. It obtains dynamic vibration characteristics of each cluster area, and then carries out structural modal analysis (e.g., mode shape). It proves that the clustering for WSN-based SHM should meet some extra requirements for modal analysts.

In the SHM perspective, although C-SHM is distributed and shown to outperform centralized approaches, it carries out excessive modal analysis at the cluster level. In the WSN perspective, C-SHM is resource-consuming, where a CH needs a lot of computation, delay, and transmission, due to such modal analysis. When we deploy a homogeneous WSN, such a CH may be the bottleneck, and may fail before a period of monitoring is over. We overcome these drawbacks. Section 5 further elaborates on these.

5.1.1 Cluster as a Subgraph

We consider each cluster as a subgraph G' of the WSN $G = (V, E)$ (where V is a set of deployed sensors and E is the set of edges) as to detect repairing points (RPs) on the deployed WSN, since each cluster with the primary sensors may be fault-prone or weakly connected ($k \geq 1$). Some sensors may not become parts of the cluster. After clustering, our objective is to detect the RPs in the WSN, and provide fault tolerance for the RPs and for the data packet-loss in each cluster. The level of fault tolerance is the failure of up to $k-1$ sensors, which means to achieve a k-connected cluster. The sensor fault tolerance has to be achieved by all independent CHs and sensors (i.e., cluster members).

5.2 Backup Sensor Placement at Repairing Points

In this section, we present the BSP algorithm for FTSHM. Let n denote the number of sensors in a cluster, and a primary sensor denote a 'sensor' hereafter.

Algorithm 1 BSP - Backup Sensor Placement

```
    input: $k, G' = (V', E'), G' \subset G$
    output: Placement of a set $B$ of $R$ backup sensors
Step 1: for each $G'$ do:
        call BSP1 ($B_1 \leftarrow B$) //Find separable points and
            place backup sensors
        call BSP2 ($B_2 \leftarrow (B - B_1)$) //Find critical middle
            points and place backup sensors
        call BSP3 ($B_3 \leftarrow (B - (B_1 + B_2))$) //Find isolated
            points and place backup sensors
      $R_r \leftarrow |(B - (B_1 + B_2 + B_3))|$ //remaining backup
      sensors
Step 2: if $R_r > 0$ then//still available backup sensors
        for each $G'$ do:
            for each location $l_i$ do:
                if there is no backup sensor placed at $l_i$ then
                  $l_i' \leftarrow l_i(p), p \in P$ //primary sensor location
                  if ($R_r \geq 1$) then //at least a backup sensor
                    place a backup sensor at $l_i'$
Step 3: Establish $k$ Connectivity
        call $k$-Connectivity-Recovery ($k$)
```

The placement of backup sensors is performed through each cluster. BSP algorithm is relatively simple: finding locations to place the backup sensors, and improving unstable or weak k-connected clusters into strongly k-connected clusters. First, BSP algorithm detects all of the repairing (or failure) points (RPs) step by step. The possible RPs are separable points, critical middle points, and isolated points in the WSN. Then, the algorithm places backup sensors until all RPs are found, or all backup sensors are placed through three sub-algorithms, namely, BSP1, BSP2, and BSP3. All three algorithms call another algorithm, *Search-and-Place*, for finding locations around each RP. We will subsequently describe BSP1, BSP2, and BSP3.

In the BSP algorithm (see Algorithm 1), there is a set B of R backup sensors, k is a degree of connectivity, and all clusters of the network are inputs. The output is the placement of R

backup sensors, and a strongly k-connected network. The BSP algorithm has three steps. In Step 1, BSP algorithm first calls the three sub-algorithms (i.e., BSP1, BSP2, BSP3) that are used to detect RPs. Each of them calls an algorithm for placing backup sensors at the RPs. When running any of the three algorithms, if there is no RP in any cluster, the algorithm stops searching and goes to the next algorithm. When the three algorithms are executed, Step 2 continues. When Step 2 is over, BSP algorithm goes to the next cluster. When placement of all of the given backup sensors is over, the BSP algorithm terminates. Let $|B_1|$, $|B_2|$, and $|B_3|$ be the number of backup sensors to be placed during the BSP1, BSP2, and BSP3 algorithms' runtimes, respectively, where $R_r \leftarrow (|B_1| + |B_2| + |B_3|) \leq |B|$.

We consider Step 2 as an option. If there are still some backup sensors available to be placed, i.e., $R_r > 0$, we can place them or save them. We think that the WSN can be sufficiently connected after placement of the backup sensors. However, there may remain some backup sensors to be placed, since the total number of RPs can be less or more than R. If R is less than the number of RPs (after placing the backup sensors), we discard finding RPs. Step 2 checks whether there are still backup sensors available or not.

If an SHM user does not wish to place more backup sensors, Step 2 can be skipped. However, we did not skip Step 2 in our evaluation. We think that the same physical sensor can feed into two or more sensors, depending on the availability of backup sensors. Step 2 checks all the sensor locations through each cluster one by one, and counts how many backup sensors are placed at each location. If a location is still with only one sensor (i.e., no backup sensor is placed yet), a backup sensor is placed at the location. If two sensors are already placed at a location, including one backup sensor, we skip the location. The placement continues until all the backup sensors are placed, i.e., $R_r = 0$.

k-Connectivity Maintenance: In Step 3, BSP algorithm calls a connectivity maintenance algorithm, *k-Connectivity-Recovery*, which starts with a cluster (all the clusters are static, but the number of sensors may change due to sensor faults). The purpose of this algorithm is to improve connectivity of the WSN in an event of sensor failure, or connectivity degradation. If all the connections belonging to a failed (or removed) sensor fail, we still require the improvement of the current connectivity to k-connectivity. The value of k can be fixed, such as $k = 3$. For the case of $k \geq 2$, the minimum weight k-connected cluster is known to be NP-hard. Related k-connectivity algorithms and theorems can be found in the literature [27], [28].

5.2.1 BSP1: Finding Separable Points as RPs

Suppose that a cluster is weakly k-connected, or sensor connectivity in a cluster area is unstable. There is one most-likely asked question: is a cluster separable? This means, are there one or more separable points in a cluster. A separable point or sensor of a cluster is an RP, that is, *the only connecting point of several other sensors, which is critical to communication, and whose removal results in a disconnected cluster* (see the half-black or dashed line in Fig. 2). In other words, a sensor ω is said to be a separable sensor *when (i) ω is in every path from a sensor u to a sensor v, or (ii) ω itself is a separable point*. In this case, we allow to set u or $v = \omega$, e.g., v_9. Indeed, sensor failure is expected to be quite common due to energy depletion. Failure of one or more sensors reduces the number of multi-hop paths, and also can cause a cluster—which has actually not failed—to become separated from the rest.

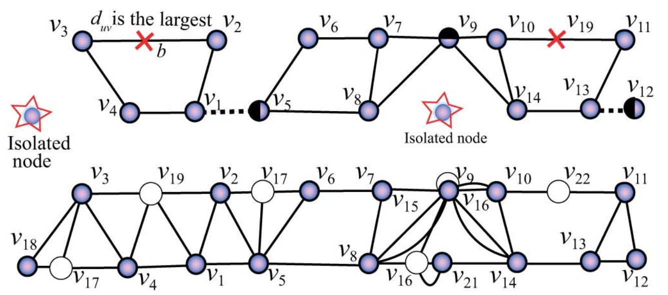

Fig. 2. Three types of RPs available in a WSN deployed on a structure (e.g., bridge, building): (a) finding separable points as RPs, half-black vertices are RPs, i.e., there is only one connection from v_1 to v_5, v_{13} to v_{12}, and v_9 is a connector of multiple neighboring nodes, (b) finding critical middle points as RPs (cross-mark), and (c) finding isolated points (star mark).

Algorithm 2 BSP1–Find Separable Points

```
Step 1: Find separable points $RP(l_i)$, Given $G' \subseteq G$
    • generate a spanning tree in $G'$
        ◦ sensor $\nu$ is not a separable point if it has no successor
          OR if each of its successors admits a descendant
          who has a back edge to an ancestor of $\nu$
        ◦ particular case: the root is a separable point
          if it has more than one successor in $G'$
Step 2: **if** (available $RP(l_i)$)
    if $((M - N) > 0)$ then
        **call** Search-and-place $(RP(l_i))$
    **else** //not available $RP(l_i)$
        discover a neighbor nearest by $RP(l_i)$
        at one hop with the increasing $R_{sr}$
```

The problem is to place a subset $B_1 \subset B$ of backup sensors, where $|B_1| \geq 0$, such that (i) the resultant network is without any separable sensor or link, and (ii) every cluster is k-connected. Finding these RPs, and placing backup sensors around the RPs, guarantee, the monitoring of every sensor location. This is done by receiving local mode shape results from all of the sensors. We list three occurrences of separable points as RPs in the network (as shown in Fig. 2): (1) v_1 and v_5 are RPs; (2) v_9 is a RP; (3) v_{12} is a RP.

BSP1 (see Algorithm 2) is given to find all such RPs through a cluster to another cluster. The running time is also different in different clusters, depending on the number of sensors n and RPs in the respective cluster. Obviously, some of the clusters are not qualified as connected clusters, since edges are limited in some cases. In fact, by running BSP1, some sensors are still poorly connected or not connected at all to other sensors. To tackle this, we use BSP2 and BSP3 algorithms. By generating the spanning tree, BSP1 typically requires the traversing of an entire cluster. Hence, the time complexity of

BSP1 is $O(n)$, linear in the size of the cluster in the worst-case, where n is the number of sensors in a cluster. In monitoring both n vertices and e edges, the complexity of the overall performance can be given by $O(n + e)$.

5.2.2 BSP2: Finding Critical Middle Points as RPs

A middle point between two sensors u and v of a cluster is an RP *which is with the longest and irregular transmission distance (d_{uv}) and the link between u and v is vulnerable* (see cross marked in Fig. 2). In order to achieve balance in d_{uv} and to receive data packets reliably and continuously, we place a backup sensor (b) in between u and v. The BSP2 (see Algorithm 3) finds such RPs in each cluster. The following two cases are considered in BSP2:

Algorithm 3 BSP2–Find Critical Middle Points

```
Step 1: Find the critical middle points in $G'$ of $G$,
        $G$ includes $P + B_1, B_2 = B - B_1$
   **for** each pairs of sensors $u$ and $\nu$ **do:**
     check $d_{uv} \leq 2d$
     **if** (available such $d_{uv}$) **then**
         calculate all $RP(l_i)$ at the middle point $(x, y)$
Step 2: **if** (available $RP(l_i)$) **then**
          if $((M - N) - |B_1|) > 0$ then
             call Search-and-place $(RP(l_i))$
        **else** //not available $RP(l_i)$
             discover a neighbor nearby $RP(l_i)$
               at one hop with the increasing $R_{sr}$
```

- d_{uv} is the longest and most irregular, i.e., the communication between u and v is not reliable. In some cases, even v may not be reachable. If v is a CH, data packets transmitted by u to v may be lost, or the receiver of v fails in the presence of physical or environmental interference.
- There may be obstacles, e.g., large metallic objects on the structure.

We use the communication circle, which is not fully circle-shaped, when sensor placement is on a bridge or building structure. Suppose that both sensors u and v are in the communication circle of another sensor. We denote d as the average reachable distance between sensors u and v. We have $d_{uv} \leq 2d$. When we place a backup sensor ($b \in B_2$) in the proximity of the middle point on the line $v_3 \rightarrow v_2$ (see Fig. 2), we have $d_{v_3b} = d_{bv_2} \leq d$. That is, sensor v_3 can easily communicate with sensor v_2 via b; the chance of data packet-loss is reduced. We have a subset $B_2 \subset B$ of backup sensors that can be placed during BSP2 runtime. BSP2 takes $O(n + e)$ time in the worst case to find critical middle points in each cluster.

5.2.3 BSP3: Finding Isolated Points

When a sensor does not have *a path or communication to another sensor in any cluster, or may receive broken messages*, the sensor is an isolated sensor. We consider the sensor location as an RP. If there is only one potential sensor in a cluster, it would be automatically chosen as an isolated sensor and a CH.

Our investigation on EFI-based sensor deployment shows that a small portion of sensors do not join the clusters, as shown in Fig. 2 (star marked). Normally, a long distance single-hop transmission is not reliable. This is because the exclusion of a small number of sensors is ignored by the cluster, which is isolated in the WSN. They may not provide the exact mode shape of their locations. For collecting mode shapes from all of the sensor locations, including the isolated one, we should guarantee that all sensors are strongly connected and data packet-loss is reduced. Thus, we require placing backup sensors to support all of the isolated sensor locations. We explain two facts for finding isolated sensors via BSP3 (Algorithm 4):

Algorithm 4 BSP3–Find Isolated Points

```
Step 1: Find isolated sensors in $G'$ of $G$
          $G$ includes $P + B_1 + B_2, B_3 = B - (B_1 + B_2)$
     A sensor $\nu$ checks its recorded table, whether there is any
          sensor with broken messages or not in a cluster
Step 2: **if** a sensor $\nu$ discovers an isolated sensor **then**
          issue a "connection notification message" to the
            sensor by increasing its transmission range
          **if** $\nu$ receives a reply from the sensor **then**
              find $RP(l_i)$ close to the sensor location
            **if** (available $RP(l_i)$) **then**
              if $((M - N) - |(B_1 + B_2)|) > 0$ then
                 **call** Search-and-Place $(RP(l_i))$
Step 3: **if** there is no isolated sensor or
            backup sensors are already placed **then**
            STOP searching // $G'$ is already connected
          **else** go to Step 2 //there is still an isolated sensor
Step 4: Find the one among the isolated sensors,
              which is the closest to an exisitng sensor
            connect it with the existing sensors
        Otherwise **return** to Step 2
```

- A CH cannot directly communicate with an isolated sensor because of an obstacle (or another reason). A CH or some sensors receive some low weight messages directly, or via some other sensors in the cluster. The isolated sensor may communicate with the other sensors occasionally; this connection is called *sporadic*.
- An isolated sensor may communicate directly with the BS. In that case, the isolated sensor runs out of energy quickly, due to using R_{lr}.

Hence, we place a backup sensor through BSP3 between a connected sensor and the isolated sensor. If a sensor fails, a backup sensor works accordingly. BSP3 mainly begins with n in a cluster. It connects any arbitrarily chosen sensors to its nearest one, which has information stored in its memory about an isolated sensor. The procedure continues until all isolated sensors are connected. The sensors in a cluster just check their memory for any irregular or low-weight messages. Also, the sensors in the boundary may essentially have information about an isolated sensor.

If we run BSP2 in the cluster for checking each connection, the procedure in BSP3 (Algorithm 5) can be executed in, at most, $O(n^2)$ time. However, comparatively, it is low many times, depending on how many isolated sensors are near to a sensor. If the number of sensors that have the isolated sensor information is n_I, the procedure in BSP3 can be executed in $O(n_I^2)$. Obviously, $O(n_I^2) \leq O(n^2)$. However, n_I may vary from one location to another, one structure to another (i.e., bridge to building, or others). $n_I = 0$ in many cases, i.e., $O(n_I^2) \geq 0$. $n_I = 0$ means that there are no isolated sensors found around a cluster. A total of $|B_3|$ sensors can be placed during BSP3 runtimes.

Algorithm 5 Search-and-Place $(RP(l_i))$

```
1: $M_{rest} \leftarrow$ Remaining locations after placement of each
   sensor placement at a $RP(l_i)$, initially, $M_{rest} \leftarrow M - N$
2: while $(M_{rest} > 0)$ do:
3:   $M_{asc} \leftarrow$ rank the locations of $M_{rest}$ based on EFI values
4:   find $RP(l_i)$ locations from $M_{asc}$ with larger EFI values
5:   for $RP(l_i) \in \{\text{neighbor locations of } l_i \text{ within } d_{uv}\}$ do:
6:     $RP(l_i) \leftarrow \text{select locations from } M_{asc}, //i, r = 1, 2, \dots$
7:   return $R_r$ //the number of locations found
8:   if $(|B| \geq 1)$ then //available sensors
9:     if $(R_r| \geq 1)$ then //available locations
10:      for each location at $RP(l_i)$ do:
11:        place a backup sensor at each $RP(l_i)'$
12:    else //not available locations
13:        place a backup sensor at $l_i$
14:          //existing sensor location, which is a $RP$
```

5.2.4 Placing Backup Sensors at the RPs

In this subsection, we provide Algorithm 5, called *Search-and-Place*. This is executed by BSP1, BSP2, and BSP3 algorithms after having found RPs, and placing backup sensors at or around the RPs. We have $M - N$ feasible locations, and a set B of R given backup sensors (line 1). The locations of the sensors can be the same or around the location coordinate (RP(l_i)) of RPs. The *Search-and-Place* checks whether there is an available sensor location from $M - N$ according to the EFI values. After the deployment of each backup sensor, the algorithm sorts the rest of the locations of $M - N$ (line 3). This is because, when a sensor is placed at a location, the location may not have a larger EFI value.

We need an optimal (or near optimal) location around an RP. However, there may be several sensor locations around an RP. In that case, we prefer the location which is along the RP. If there are several locations available along with an RP, then the location with the larger EFI value is a better choice (line 5). Then, the output (line 7) can be the locations near the RP. R_r denotes the number of locations found near the RP. R_r can be zero to many, which means that there is a chance of having more than one location, or no locations available. If there is a backup sensor available, but no location near an RP, the algorithm places a backup sensor at the same sensor location (which is an RP) (line 13). This is because this algorithm is called by all BSP1, BSP2, and BSP3 algorithms whenever a backup sensor needs to be placed at a specified location. Hence, the computation time for this algorithm is equivalent to at least $O(M - N)$, where $M - N$ is the number of feasible locations.

6 Energy-Efficient Online SHM

In this section, we select another fundamental problem in SHM—detecting structural damage, and try to tailor it from a WSN point of view. As described in Section 3, according to civil engineering, mode shapes are global parameter, computed at the BS based on raw data collection in existing wired, and even some wireless, sensor-based systems. Since detecting damage via mode shape analysis requires high-rate data collection, each node produces too much data for its own and other radios [19]. Given the constraints in WSNs, it is particularly difficult to get a possible damage detected by the BS. We present an energy-efficient online damage indication algorithm, called *Damage-Indicator*, to optimize WSN resources and make the WSN easier to apply for SHM.

6.1 Damage-Indicator Algorithm

We attempt to implement mode shape computation on each sensor in a decentralized manner (see Algorithm 6). As described in section 3.1, it is seen that a global mode Φ_k has an element corresponding to each sensor node. This implies that we can also analyze a local mode at each sensor that is illustrated in Algorithm 6. Advanced sensor platforms, such as Imote2, which we use in this work, easily compute it; we can even scale down the Imote2's computing power [33] to compute each mode shape. A salient feature of the algorithm is the partitioning of a global mode shape between the CHs and the BS, and further partitioning between a CH and each wireless sensor. Each sensor extracts the peak frequencies from the acquired set of captured frequencies. We make the case for our network model in which each sensor stores data locally, i.e., all sets of frequencies are stored in an external flash or in the local memory. We utilize a database query interface to the specific portion of data, if needed. Imote2 has 32MB local memory. All of the acquired frequencies remain stored in the memory until a confirmation is sent, from the BS through a CH, that there is no query request (e.g., no damage). This significantly reduces the communication load, and hence, energy consumption in exchange for moderate computation costs on each sensor.

In Algorithm 6, there are two levels: one runs at the node and another one runs at the CH. At the node level, each sensor node runs five steps.

1) Each sensor takes measurements and identifies the initial set of frequencies ($f_i, i = 1, 2, \dots, k$), which are achieved by converting measured accelerations to the frequency domain using a fast Fourier transform (FFT) [6], [8].
2) Arrange a consistent set of frequencies, which are first temporarily stored in the buffer.

3) The maximum number of largest peaks are specified as p, which can be identified by scanning f_i (see the Peak-Picking method in Appendix C) [34]. Through this, the amount of given sampling data points (e.g., 2048 bytes), are transformed into x floats, which reduces to $x/4$ floats in Step 3. If less than p peaks are found, zeros will be returned in place of the missing peaks.
4) The mode is assembled by finding the closely related modes and saving it as ϕ into the buffer.
5) In the case of i-th sensor S_i, Φ_i is computed by repeating Steps 2 to 4, and buffering on the local memory. Then, the node transmits it to the CH. Here, x reduces to I floats, i.e., each final local mode shape is up to a maximum of 20 bytes.

Algorithm 6 Online SHM Algorithm (Damage-Indicator)

```
Input: Given a WSN deployed on a structure and reference mode shapes;
Output: Damage indication (if any);
  At the node level:
    for each individual sensor node $S_1$ do:
      (Upon the data measurement, $y_i[t]$)
        Step 1. Conduct FFT analysis on the measured data;
            // $x$ integer
        Step 2. Identify frequency response set, $F_i$;   // $x$ float
            //$x$: # of samples, $I$: # of identified frequencies
        Step 3. Scan $F_i$ for 'Peak Picking',
            $f_i = [f_1, f_2, f_3, \dots, f_n]_1$; // $x/4$ float
        Step 4. Assemble mode shape, $\phi_i$;   // $(x/4)/4$ float
        Step 5. Identify mode shape, $\Phi_1 = [\phi_1, \phi_3, \phi_3, \dots, \phi_n]$
        and buffer it;   // $I$ float, $x >> I$
        Transmit it to the CH;
    end
At the CH level:
    for each CH do:   // any cluster of the WSN
        (Upon the reception of all mode shapes from all of the
                sensors in the respective cluster
                $S_1 = S_1, S_2, S_3, \dots, S_{n-1}$)
      Estimate the mode shapes in the cluster,
          $\Phi_i[\phi_1, \phi_3, \phi_3, \dots, \phi_c]$;
      Get the reference mode shape for the cluster;
      ComputeDamageIndication{   //if there is a significant
          change
        if Sensor $S_1$.indication = true {
        Compare $S_1$.indication with $S_{n-1}$.indication
            in the cluster;
        Transmit the indication toward the BS;}
        else transmit $S_1$'s working status;} // e.g., normal,
        faulty
    end
```

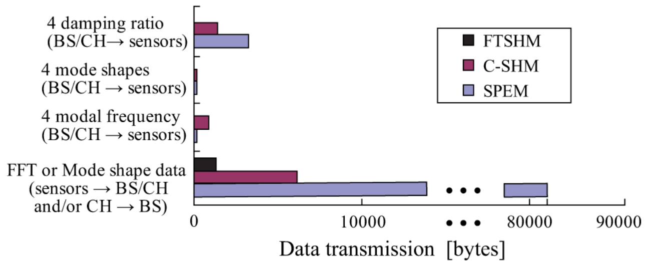

Fig. 3. Summary of wireless data transmission analyzed by the results in different WSN-based SHM approaches in the 10-sensors case.

At the CH level, upon reception of all of the mode shapes from the sensors S_i, a CH assembles all of the mode shapes and finds significant changes in each mode shape sent by each sensor. We assume that each CH is aware of a *reference mode shape* (e.g., $\Phi_4 = \{3.12, 2.434, 4.23, 3.643\}$ of four sensor locations, see Definition 3) of a cluster. Through a simple comparison between the identified and reference mode shapes, the CH provides an indication about damage.

If there is an indication of a possible damage, a CH transmits the indication to the BS; otherwise, it just maintains connectivity with the BS and the sensors. The BS receives indications from all of the CHs and is able to know the health of the whole structure. After analyzing, if the BS determines that there is possibly damage, the BS can query the corresponding cluster or the sensors for detailed mode shapes, even for all sets of frequencies. In this way, if there is *no damage indication* (because we think that damage is *a kind of event that rarely occurs in a structure*), the amount of data is reduced before transmission. We utilize such a query-interface-based system suggested in [35].

6.2 Performance Analysis

Through the *Damage-Indicator* algorithm, we control the communication burden in the WSN. The control of the amount of wireless transmission and the amount of bandwidth needed for the transmission in FTSHM, SPEM, and C-SHM can be seen in Fig. 3. Fig. 3 analyzes the results of 10 sensors that are deployed on a real structure. In FTSHM, the sensors are placed on different locations of the structure. The experiment is performed over a 7-hour period. Each node is with a 3-axes accelerometer, sampling at 560 Hz with a 140 cutoff frequency (see Section 8 for more details).

SPEM finds 11 sensor locations and places 16 sensors, and collects acceleration data at 560 Hz for 24 hours. Each sensor transmits their raw FFT data to the BS directly for modal analysis. 4,096 data points are transmitted from each sensor, resulting in a total of 81,920 bytes being transmitted (each point is a 2 byte number). If the BS communicates to each sensor for raw FFT using SPT (shortest path tree routing in SPEM), an additional 3,200 bytes are transmitted, bringing the total number of bytes to 85,440.

In C-SHM, FFT-based modal analysis is performed at the CH and all of the mode shapes in a cluster are transmitted to the BS. It performs much better than SPEM. The same final results of SPEM can be obtained by transmitting a total of 8,582 bytes of data, i.e., about 10 times reduction. This broadcasting can be reduced to require only 640 bytes of transmission in FTSHM. This is in the case, when every sensor makes a decision and produces a mode shape through Algorithm 6,

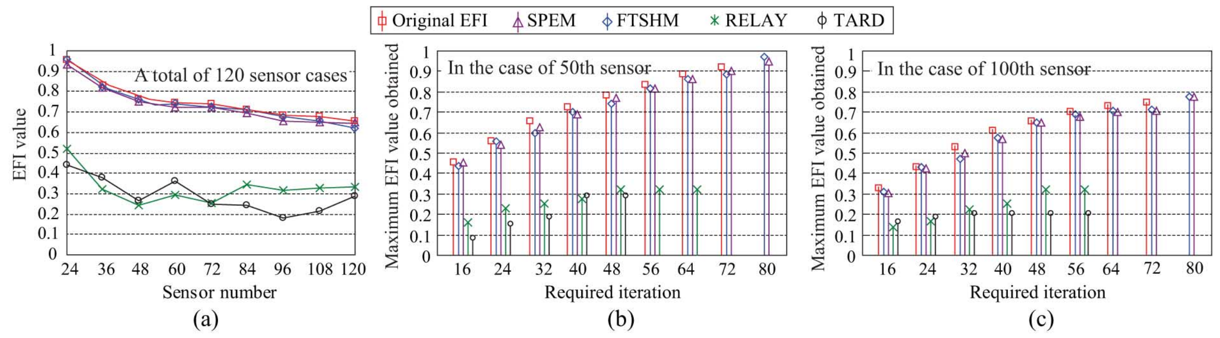

Fig. 4. Performance on the location quality: (a) variant EFI values vs. the number of sensors; (b) maximum EFI values obtained by the 50th sensor; (c) maximum EFI values obtained by the 100th sensor.

and transmits to the CHs. Note that there are two clusters with four and six sensors, respectively. The CHs transmit an indication of damage with a final mode shape of the cluster. However, the amount of data transmission can be increased when there is a damage indication or fault in the cluster. This is because the BS may request a query for the transmission of all data sets to the cluster or the sensors which indicated the damage. Bearing in mind the *damage* as a rare event in the structure, the reduction of such data transmission in a WSN is a significant result.

7 Performance Evaluation

7.1 Method and System Parameters

We conduct simulations using real data sets to demonstrate the effectiveness of FTSHM. Both the primary and backup sensors' deployment algorithms are performed through a high-rise building structure model, based on the GNTVT [10]. The data sets are collected from the GNTVT SHM system and are injected into the system by the sensor data acquisition module.

EFI-based sensor placement in SPEM [11] provides a module as an open package (both MATLAB toolbox and C++ version). In our simulation, all geometric and material properties are accurately adopted without any changes, but we improve the sensor placement module by adding a BSP algorithm and fault tolerance sub-modules. SPEM restricts the module to m locations, while we relax this restriction and just use the EFI method for the N primary sensors.

The tunable parameters in FTSHM are as follows. (i) We consider $m = 120$ in all three approaches for suitability. In FTSHM, $N = 90$ and $R = 30$, while $N = 120 (= m)$ and $R = 0$ in both SPEM and RELAY. (ii) k can be 2 to 5, or more; we consider $k = 3$ in simulations. (iii) We set $R_{lr} = 80m$ and $R_{sr} = 30m$ for communication (modeled in Section 4.1). (iv) The communication cost is normalized to 100 (see (4)). (v) The WSN lifetime (T) is normalized to 1 for all approaches (modeled in Section 4.3). Low energy consumption of sensors is achieved via a work/sleep mode along with a low duty cycle of 2%. This is usually feasible for a WSN, since many SHM applications only require the collection of 5 to 10 minutes of data a day to reduce power consumption. The duty cycle for a CH's radio is set to a higher amount (3%) for additional communication, if needed.

Besides the performance comparison of data transmission in Fig. 3, the performance of FTSHM in more aspects (e.g., location quality, lifetime, communication cost, mode shape identification, and clustering) is compared. We implement another three approaches. SPEM [11] and RELAY [29] are described in Section 2. Another solution to Traffic-Aware Relay Deployment (TARD for short) is suggested with an objective to maximize lifetime for a data collection WSN. It shows that the general form of the deployment problem is difficult. The only existing connectivity-guaranteed solutions cannot be directly applied in some applications like SHM. It then develops algorithms for discrete relay node assignment, together with local adjustments that yield high-quality sensor deployment.

Fault Injection. We inject faults into the WSN to investigate to what extent T prolongs, and what communication cost (for recovery from sensor faults and data packet-loss) is required. f_r denotes the fault injection in a random manner, estimated by *the percentage of the number of faulty sensors to the total number of sensors given*. However, if one wishes, f_r can also be given by switching off, sensor debonding fault, minimizing energy, etc. After a fault occurs, the *k-Connectivity-Recovery* algorithm runs on the remaining sensors for recovering from the situations and establishing the minimal connectivity to the required k-connectivity.

7.2 Results

We first analyze the sensor placement performance of different approaches in Figs. 4(a) to 4(c). We summarize the results of all of the sensors in different cases. We observe that the placement quality decreases as the number of sensors increases. In FTSHM, in the case of backup sensors, the placement quality is slightly affected, compared to the baseline engineering EFI method. But in 60-sensor, 84-sensor, 96-sensor, and 108-sensor cases, FTSHM performs better than SPEM.

Another two approaches, REALY and TARD, show low location quality. Their poor performance proves that deployment should satisfy application demand from SHM, although they both require a minimum number of iterations. It is also evident that, as a location with minimum quality deleted from the candidate set, the individual location quality associated with the remaining locations may vary accordingly. In Fig. 4 (b) and 4(c), we see that the 50th sensor location reaches the maximum EFI value at the 81th iteration in FTSHM, while it is at the 72th iteration in SPEM. The EFI value decreases with the number of iterations.

Second, we study T vs. m in the WSN. The simulation results are shown from Fig. 5(a) to Fig. 5(c), with different

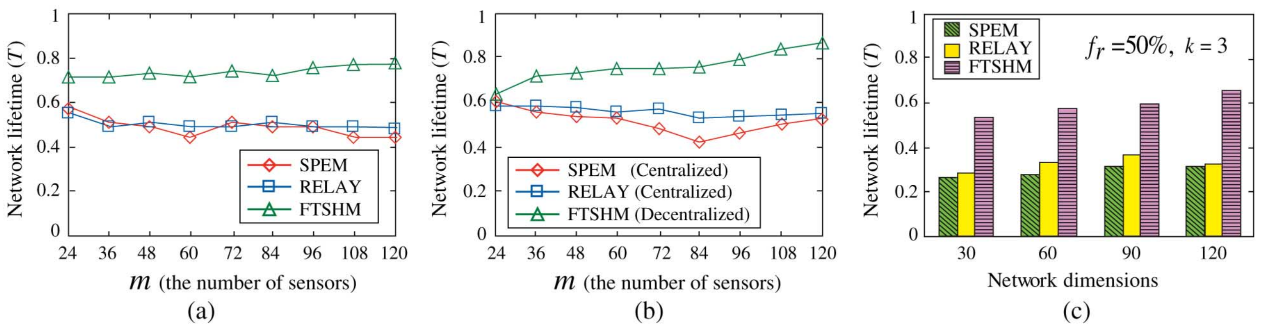

Fig. 5. WSN performance achieved in simulations: (a) T vs. m under normal conditions; (b) T vs. m under measurement, computation, and transmission; (c) T vs. different network dimensions under $f_r = 50\%$ and $k = 3$.

settings. We obtain that FTSHM shows better (roughly 35% to 65%) performance than both SPEM and RELAY. When the number of sensors increases, T increases in FTSHM, while it is stable, even with a slight decrease in both SPEM and RELAY. Looking into the details, placing additional sensors anywhere does not give the desired results for SHM, and also affects data traffic and congestion in the WSN. As a result, data traffic may not be evenly distributed by simply placing more sensors. In both SPEM and RELAY, $N(= m) = 120$. In FTSHM, $N(= 90)$ are placed at the optimal location with high EFI values, while $R(= 30)$ are placed at the optimal (or near optimal) locations by detecting the RPs so that R_{lr} is minimized. R_{sr} is used inside a cluster. These are some reasons that T in FTSHM increases in all cases, including in the presence of sensor fault in Fig. 5(c); $f_r = 50\%$. The impact is that when sensor faults occur, the remaining sensors reestablish k-connectivity.

Another reason is that the mode shape computation and decision making are centralized in both SPEM and RELAY. We restrict unnecessary transmission if there is *no damage indication*. It is seen that RELAY performs better on the WSN lifetime than SPEM. Recall that RELAY shows the worst performance on sensor placement.

Under the same set of simulations, Fig. 6(a) shows that the remaining sensors suffer from 50% sensor faults to reestablish k-connectivity. The number of sensors reduces in each cluster. Both SPEM and RELAY show significant points of failure, namely, 4th, 7th, 11th, and 13th clusters' failure, i.e., the network partition appears. It implies that the data about the health status of the structure, e.g., mode shape, cannot be collected from those failed sensor locations.

FTSHM is still robust under such a sensor fault rate. We can see that two sensors remain working in the fourth cluster; we still can expect to achieve mode shape results for the locations. Fig. 6(b) plots the communication cost when f_r is up to 50%. Each of the plotted values are normalized first by f_r. Both SPEM and RELAY frequently maintain the connectivity in the clusters to recover from faulty sensors. The communication cost of SPEM is much higher than RELAY.

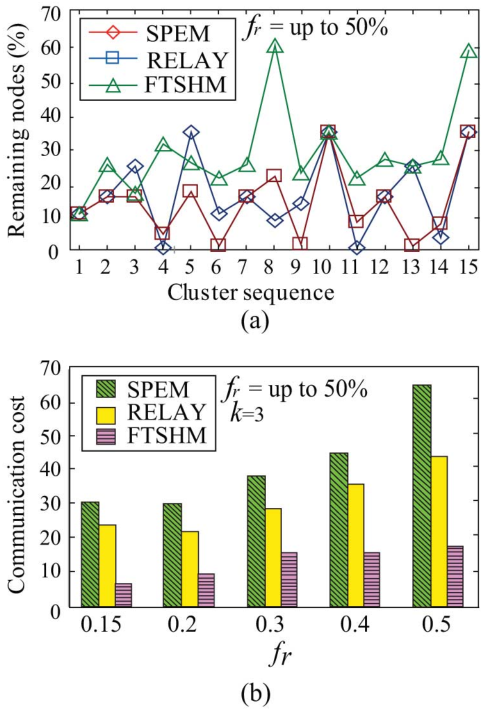

Fig. 6. WSN performance under sensor faults and fault-tolerance: (a) the number of remaining sensors in each cluster; (b) communication cost when $f_r = 50\%$.

8 Proof-of-Concept Experiments

As a proof-of-concept experiment, we have conducted a field experiment on the *Lee Shao Kee* (LSK) building located at the Hong Kong PolyU campus (see Fig. 7). The objective of this deployment is to validate the feasibility of placement performance, fault tolerance, and decentralized data processing in SHM.

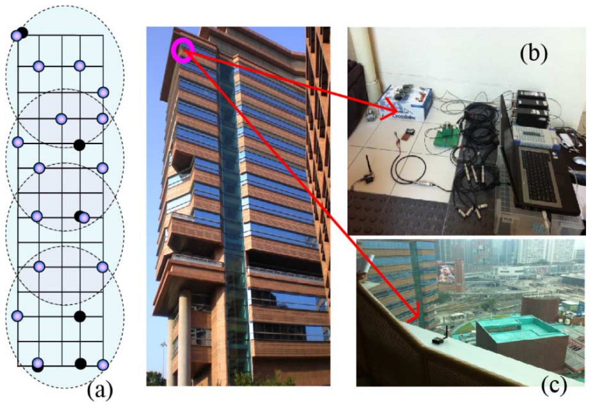

Fig. 7. Sensor deployment on LSK building: (a) an example of FEM model where 16 primary and 6 backup mote (black nodes) are placed; (b) BS mote location; (c) the placement of a sensor.

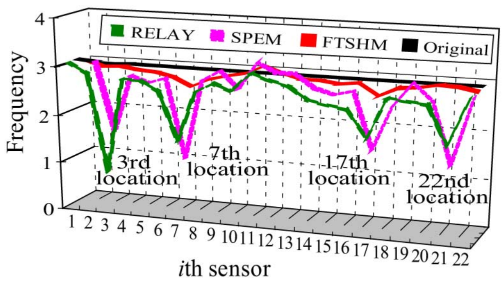

Fig. 8. Impact of sensor faults on SHM performance: identified affected mode shape under sensor faults.

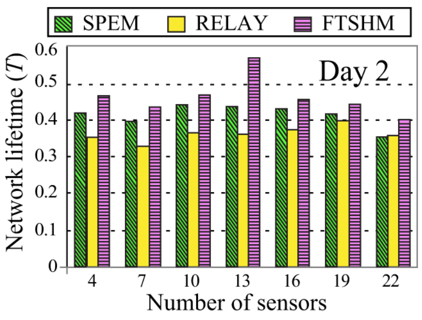

Fig. 9. Impact of sensor faults in the network performance: T vs. the number of sensors.

The structure is instrumented by deploying motes, called *SHM mote*, integrated by off-the-self Intel Imote2 sensors [33]. TinyOS 2.1 [36] runs on the Imote2. According to our proposed algorithms, on the first day of deployment, we select 22 locations for 22 motes that seem to capture the overall vibration frequency and mode shape without any sensor fault. There are at least 220 locations ($= M$) on the structure computed by the FEM model of the structure [5], [8].

On the 2nd day, we first place 16 motes ($= N$), according to the EFI method. We find location coordinates of 6 RPs in the WSN, and find appropriate locations out of the remaining locations (204 locations) for the 6 backup sensors ($= R$). We inject faults into 2 sensors placed on the 5th floor and 7th floor, remove one sensor from the 11th floor, and inject sensor debonding fault into a sensor on the 13th floor.

8.1 Experimental Results

We only analyze the results of the mode shape identification and network lifetime under the sensor fault condition. Fig. 8 demonstrates mode shapes and reveals some severe changes. These changes are not actually affected by damage, but by the sensor faults in the WSN of SPEM—where there are no backup sensors available at the locations of the failed sensors, and no recovery solutions from the situations. This indicates that in the case of the WSN, the engineering deployment methods bring more chances to WSNs to be prone to faults, where communication distance varies greatly. In contrast, RELAY has better performance than that of SPEM at the faulty sensor locations, but worse performance at some other locations. This is due to random deployment. We note that the distorted mode shapes in both SPEM and RELAY may lead to an *indication/alert* for damage detection (i.e., false-negative detection). A closer look reveals that the identified mode shape is slightly distorted ($< 6\%$) in `FTSHM`. This does not have much of an impact in SHM. The distortion is more than 35% in both SPEM and RELAY. The results show that `FTSHM` is able to overcome the situations.

Fig. 9 illustrates T under sensor faults. It is seen that T decreases (around 4% in SPEM, and around 6% in RELAY) after each sensor fault or failure. It specifies that T prolongs in `FTSHM`, compared to SPEM and RELAY. When a sensor fails, which is an intermediate sensor on the shortest path in SPEM, the number of packet retransmissions largely increases.

A detailed description of the system implementation, and more performance results, can be found in Appendix D of the supplemental material.

9 Conclusion

In this paper, our intention was to demonstrate a new way of incorporating the requirements of both WSN and SHM, and to make use of traditional engineering methods in the WSN. We found that it is worthwhile to place a small number of backup sensors around the repair points in the WSN to have a better performance. We believe that such an idea (of the backup sensor placement) can also be used in generic WSN applications. Besides, we proposed an SHM algorithm exploiting sensor-decentralized computing in the resource-constrained WSN. Through extensive simulations and a real implementation using integrated Imote2 sensors, we validated the effectiveness of our approach. The validation shows that structural health monitoring using WSNs can be meaningless, if the requirements of WSNs (e.g., fault tolerance, energy-efficiency) are not seriously considered.

This work leaves at least two open issues in the multi-domain research area. One issue is to develop algorithms for SHM application-specific sensor fault detection and recovery. Another issue is to develop a SHM-specific scheduling technique for the backup sensors that will wake up one or more backup sensors in the areas of interest (e.g., damaged area) in the case of a sensor fault/failure. This may help to meet both coverage and connectivity requirements in a WSN-based SHM system.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grants 61272151 and 61272496, in part by ISTCP under Grant 2013DFB10070, in part by HKRGC under GRF Grant PolyU5106/11E, HK PolyU Niche Area Fund RGC under Grant N_PolyU519/12, and in part by NSF under Grants ECCS 1231461, ECCS 1128209, CNS 1138963, CNS 1065444, and CCF 1028167.

References

[1] X. Mao, X. Miao, Y. He, T. Zhu, J. Wang, W. Dong, X. Li, and Y. Liu, "CitySee: Urban CO2 monitoring with sensors," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2012, pp. 1026–1034.
[2] G. Wang, M. Z. A. Bhuiyan, and Z. Li, "Two-level cooperative and energy-efficient tracking algorithm in wireless sensor networks," *Concurrency Comput. Practice Experience*, vol. 22, no. 4, pp. 518–537, 2010.
[3] M. Z. A. Bhuiyan, G. Wang, J. Cao, and J. Wu, "Energy and bandwidth-efficient wireless sensor networks for monitoring high-frequency events," in *Proc. 10th Annu. IEEE Commun. Soc. Conf. Sensor Mesh Ad Hoc Commun. Networks (SECON)*, 2013.

[4] M. Z. A. Bhuiyan, G. Wang, J. Cao, and J. Wu, "Local monitoring and maintenance for operational wireless sensor networks," in *Proc. 12th IEEE Int. Conf. Trust Security Privacy Comput. Commun. (ISPA)*, 2013.

[5] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *Proc. 6th Int. Symp. Inform. Process. Sensor Netw. (IPSN)*, 2007, pp. 254–263.

[6] G. Hackmann, F. Sun, N. Castaneda, C. Lu, and S. Dyke, "A holistic approach to decentralized structural damage localization using wireless sensor networks," *Comput. Commun.*, vol. 36, no. 1, pp. 29–41, 2012.

[7] K. Chebrolu, B. Raman, N. Mishra, P. K. Valiveti, and R. Kumar, "BriMon: A sensor network system for railway bridge monitoring," in *Proc. 6th Int. Conf. Mobile Syst. Appl. Services (MobiSys)*, 2008, pp. 2–14.

[8] J. A. Rice and B. F. Spencer, "Flexible smart sensor framework for autonomous full-scale structural health monitoring," Univ. Illinois at Urbana-Champaign, NSEL Report 18, 2009.

[9] J. Ko, Y. Ni, H. Zhou, J. Wang, and X. Zhou, "Investigation concerning structural health monitoring of an instrumented cable-stayed bridge," *Struct. Infrastructure Eng.*, vol. 5, no. 6, p. 2008, 497–513.

[10] *Structural health monitoring for Guangzhou new TV tower using sensor networks* [Online]. Available: http://www.cse.polyu.edu.hk/benchmark/

[11] B. Li, D. Wang, F. Wang, and Y. Q. Ni, "High quality sensor placement for SHM systems: Refocusing on application demands," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2010, pp. 650–658.

[12] X. Liu, J. Cao, S. Lai, C. Yang, H. Wu, and Y. Xu, "Energy efficient clustering for WSN-based structural health monitoring," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2011, pp. 2768–2776.

[13] D. C. Kammer, "Sensor placement for on-orbit modal identification and correlation of large space structures," *J. Guid. Control. Dyn.*, vol. 14, no. 2, pp. 251–259, 1991.

[14] M. Meo and G. Zumpano, "On the optimal sensor placement techniques for a bridge structure," *Eng. Struct.*, vol. 27, no. 2005, pp. 1488–1497, 2005.

[15] M. Z. A. Bhuiyan, J. Cao, and G. Wang, "Deploying wireless sensor networks with fault tolerance for structural health monitoring," in *Proc. IEEE 8th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, 2012, pp. 194–202.

[16] T.-H. Yi, H.-N. Li, and M. Gu, "Optimal sensor placement for structural health monitoring based on multiple optimization strategies," *Struct. Des. Tall Special Buildings*, vol. 20, no. 7, pp. 881–900, 2011.

[17] A. Jindal and M. Liu, "Networked computing in wireless sensor networks for structural health monitoring," *IEEE/ACM Trans. Netw.*, vol. 20, no. 4, pp. 1203–1216, 2012.

[18] T. Zhang, D. Wang, J. Cao, Y. Ni, L. Chen, and D. Chen, "Elevator-assisted sensor data collection for structural health monitoring," *IEEE Trans. Mobile Comput.*, vol. 11, no. 10, pp. 1555–1568, Oct. 2012.

[19] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica, "Flush: A reliable bulk transport protocol for multihop wireless networks," in *Proc. 5th Int. Conf. Embedded Netw. Sensor Syst.*, 2007, pp. 351–365.

[20] M. Z. A. Bhuiyan, G. Wang, and J. Cao, "Sensor placement with multiple objectives for structural health monitoring in WSNs," in *IEEE 9th Int. Conf. High Perform. Comput. Commun. (HPCC)*, 2012, pp. 699–706.

[21] M. Z. A. Bhuiyan, J. Cao, G. Wang, and X. Liu, "Energy-efficient and fault-tolerant structural health monitoring in wireless sensor networks," in *Proc. IEEE 31st Symp. Reliable Distrib. Syst. (SRDS)*, 2012, pp. 301–310.

[22] S. Beygzadeh, E. Salajegheh, P. Torkzadeh, J. Salajegheh, and S. Naseralavi, "Optimal sensor placement for damage detection based on a new geometrical viewpoint," *Int. J. Optimization Civil Eng.*, vol. 3, no. 1, pp. 1–21, 2013.

[23] X. Chang, R. Tan, G. Xing, Z. Yuan, C. Lu, Y. Chen, and Y. Yang, "Sensor placement algorithms for fusion-based surveillance networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 8, pp. 1407–1214, Aug. 2011.

[24] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Robust sensor placements at informative and communication-efficient locations," *ACM Trans. Sensor Netw.*, vol. 20, no. 7, pp. 1–33, 2011.

[25] S. Lee and M. F. Younis, "EQAR: Effective QoS-aware relay node placement algorithm for connecting disjoint wireless sensor subnetworks," *IEEE Trans. Comput.*, vol. 60, no. 12, pp. 1772–1787, Dec. 2011.

[26] Z. Yun, X. Bai, D. Xuan, T. H. Lai, and W. Jia, "Optimal deployment patterns for full coverage and *k*-connectivity ($k \leq 6$) wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, pp. 934–947, 2010.

[27] F. Wang, D. Wang, and J. Liu, "Traffic-aware relay node deployment: Maximizing lifetime for data collection wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 8, pp. 1415–1423, Aug. 2011.

[28] J. L. Bredin, E. D. Demaine, M. T. Hajiaghayi, and D. Rus, "Deploying sensor networks with guaranteed fault tolerance," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 216–229, 2010.

[29] K. Xu, H. Hassanein, G. Takahara, and Q. Wang, "Relay node deployment strategies in heterogeneous wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 2, pp. 145–159, Feb. 2010.

[30] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless micro-sensor networks," in *Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci. (HICSS)*, 2000, pp. 3005–3014.

[31] P. Cheng, C. N. Chuah, and X. Liu, "Energy-aware node placement in wireless sensor network," in *Proc. Global Telecommun. Conf. (GLOBECOM)*, 2004, pp. 3210–3214.

[32] S. Olariu and I. Stojmenovic, "Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2006, pp. 1–12.

[33] *Imote2 Hardware Reference Manual*, Crossbow Technology Inc., Sep. 2007, PN: 7430-0409-01.

[34] N. M. Danna and E. G. Mekonnen, "Data processing algorithms in wireless sensor networks for structural health monitoring," Master's thesis, KTH, School Archit. Built Environ., Sweden, 2012.

[35] N. Tsiftes and A. Dunkels, "A database in every sensor," in *Proc. IEEE SenSys*, 2011, pp. 316–332.

[36] *TinyOS Reference Manual* [Online]. Available: http://www.tinyos.net

Md Zakirul Alam Bhuiyan received the BSc degree in computer science and technology from International Islamic University Chittagong, Bangladesh, in 2005, and the MSc and PhD degrees in computer science and technology from Central South University, China, in 2009 and 2013, respectively. He is currently a postdoctoral research fellow at Central South University, where he is a key member of the Trusted Computing Institute of the university. His research interests include mobile computing and wireless sensor networks, cyber physical systems (CPS), and sensor data cloud computing. He was a recipient of the top-notch PhD student award and of the best paper award in IEEE ISPA 2013. He was a research assistant (RA) in Hong Kong PolyU. He is a member of IEEE and a member of ACM.

Guojun Wang received the BSc degree in geophysics, the M.Sc. and Ph.D degrees in computer science from Central South University, China. He is now Chair and Professor of the Department of Computer Science at Central South University. He is also Director of the Trusted Computing Institute of the University. He has been an Adjunct Professor at Temple University, USA; a Visiting Scholar at Florida Atlantic University, USA; a Visiting Researcher at the University fault tolerant of Aizu, Japan; and a Research Fellow at the Hong Kong Polytechnic University. His research interests include network and information security, Internet of things, and cloud computing. He is a distinguished member of CCF, and a member of IEEE, ACM, and IEICE.

Jiannong Cao received the BSc degree in computer science from Nanjing University, Nanjing, China, and the MSc and the PhD degrees in computer science from Washington State University, Pullman, WA. He is currently a chair professor and head of the Department of Computing at Hong Kong Polytechnic University. His research interests include parallel and distributed computing, computer networks, mobile and pervasive computing, fault tolerance, and middleware. He has co-authored 4 books, co-edited 9 books, and published over 300 papers in major international journals and conference proceedings. He has directed and participated in numerous research and development projects and, as a principal investigator, obtained over HK$25 million grants. He is a senior member of the China Computer Federation, a senior member of IEEE, and is a member of ACM. He is the Chair of the Technical Committee on Distributed Computing of the IEEE Computer Society. He has served as an associate editor and a member of the editorial boards of many international journals, and as a chair and member of organizing/program committees for many international conferences.

Jie Wu is the chair and a Laura H. Carnell Professor in the Department of Computer and Information Sciences at Temple University. Prior to joining Temple University, he was a program director at the National Science Foundation and Distinguished Professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He regularly published in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including *IEEE Transactions on Computers, IEEE Transactions on Service Computing*, and *Journal of Parallel and Distributed Computing*. Dr. Wu was general co-chair/chair for IEEE MASS 2006 and IEEE IPDPS 2008 and program co-chair for IEEE INFOCOM 2011. Currently, he is serving as general chair for IEEE ICDCS 2013 and ACM MobiHoc 2014, and program chair for CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). He is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.